SAPIENZA
UNIVERSITÀ DI ROMA

Word Sense Disambiguation

HOMEWORK 3 REPORT

MAKINWA, Sayo Michael
[1858908]

10.09.2019

Table of Contents

1.0 Introduction

The task is to Implement and train a system capable of performing both fine-grained and coarse-grained Word Sense Disambiguation using annotated English sentences extracted from given corpora and a given set of sense mappings, and implementing an evaluation task to test the model. The image below is a graphical representation of the full pipeline of my implementation:

Fig. 1. Graphical description of the full pipeline

2.0 The Dataset and Data Preprocessing

For this implementation, I used the **all English SEMCOR, and the SEMCOR+OMSTI corpora**, comparing the performance of the model trained on each of them. I parsed each of these corpora to extract sentences from them, and for each of these sentences, while representing the instances present in the sentences with their corresponding sense ID's.

The following are the important decisions I made to parse the corpora:

1. The input X to the model is made to consist of **raw texts** from the corpora **in their inflective form**; this is so as to retain important grammatical information in the sentences.
2. To minimize the number of output classes, the output Y of the model is built up with lemmas and sense ID's for instances.
3. All the punctuations are removed
4. I applied phrase generation (Mikolov et al. 2013) such that instances that have multiple words in their text body are maintained as single words by replacing the spaces with underscores in the input X; this is so as to maintain equal lengths of inputs and outputs and keep the problem as a simple sequence to sequence model.

3.0 Model Setup and Parameter Tuning

The baseline model, as described in the Raganato et al. paper, consists of an all words (in their inflected forms) inputs and a words (in their inflected forms) + senses output. This means that the output vocabulary, and consequently the number of classes the model needs to predict, is quite large. Now, we also really care about predicting the correct senses and not the words, so I tried a few modifications that specifically targets the reduction of the output vocabulary by removing some or all of the unannotated words, in hopes to drive up performance. The basis of the modifications done is shown in Fig. 2 where the unannotated words are marked as belonging to a single class "others". The sections that follow explain each modification.

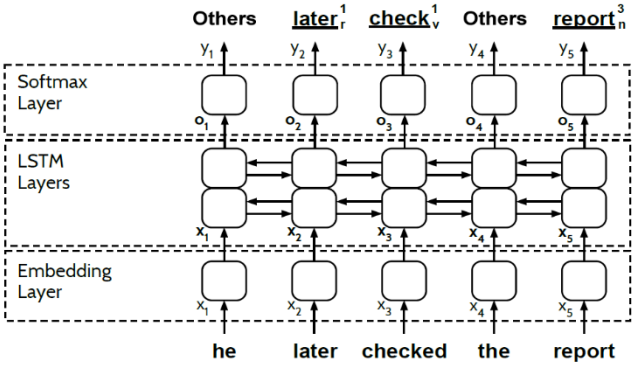

Fig 2. Basis for model modification

The following table describes the parameters used for training the models

Embedding Size	128	LR	0.001	Epochs	20
LSTM Layers	2 Bi-LSTM layers; 512 hidden size for each LSTM	Padding Size	size=30; truncating='pre', padding='post'	Data	Training: Semcor; Dev: senseval3
Optimizer	Adam	Dropout	0.3	Batch	64

Table 1. Model parameters

3.1 The Baseline Model – Model A

The structure of this model is baseline as described in the paper; the output of the model consists of lemmas of unannotated words and senses of instances, making an output vocabulary of size **49,224** which is the number of classes the model tries to learn. See Table 2 for the results of this model.

3.2 Senses-Only Model – Model B

Since we primarily care about the predictions of the senses and not necessarily the predictions of the other words, the idea here is to eliminate the classes that the unannotated words make up, leaving out only the senses, so I grouped all the words into a single class "OTHERS" (See Fig. 2), making that the only addition to the senses. This seriously dropped down the output vocabulary by almost half to **26,034**, which is the new number of classes the model has to learn. A downside to this approach is the fact that the size of things that fall under the "OTHERS" category during training becomes tremendously huge, and the model quickly learns to default to this category to increase the score, which may not be proportional to the performance of sense predictions. **For the fined-grained predictions, this model performed better than model A on the senseval3 dataset (which is my dev set), while for coarse-grained, it performed better on the senseval2 dataset** (Table 2).

3.3 Senses Plus Min-Count Words Model – Model C

Here, I tried to find a common ground between models A and B, bringing the strength of B to reduce the vocabulary, while keeping some of the unannotated words that satisfy a minimum occurrence so as to take away the disadvantage of having most of the output fall under a single category; unannotated words that do not occur up to three times were all generally classified together, same for stopwords. This took the number of output class to **34,505**. **For the fined-grained predictions, this model performed better than models A and B on the SE2 and SE15 datasets, while for coarse-grained, it performed better on the SE2 and the SE07 datasets (and on ALL on LexNames predictions)**. **It also performed better than model B on all the datasets except on SE3**.

3.4 Model C with an Additive Attention Layer

I added an additive attention layer to model C above using the equation on Fig 3. The model performed worse.

3.4 Model A with a Multiplicative Attention Layer

Further experimenting with attention layers, I added a multiplicative attention layer to model A above (slight modification over additive attention layer). While the results are not better than the base model A, it was much better than what came from using additive attention.

$$h_{t,t'} = \tanh(x_t^T W_t + x_{t'}^T W_x + b_t)$$
$$e_{t,t'} = \sigma(W_a h_{t,t'} + b_a)$$
$$a_t = \text{softmax}(e_t)$$
$$l_t = \sum_{t'} a_{t,t'} x_{t'}$$

$$h_{t,t'} = \tanh(x_t^T W_t + x_{t'}^T W_x + b_t)$$
$$e_{t,t'} = \sigma(x_t^T W_a x_{t'} + b_a)$$
$$a_t = \text{softmax}(e_t)$$
$$l_t = \sum_{t'} a_{t,t'} x_{t'}$$

Fig 3. Additive (left) and multiplicative (right) attention equations as used.

3.4 Multitask Learning

First, I added POS and Lex as auxiliary tasks to the base WSD structure while experimenting with a few ways of structuring things. However, the model always turned bad with POS always seriously overfitting to the training data. Then I switched POS for domain names, while also experimenting with different structures, as well as predicting coarse-grained from the fine-grained layer; it turns out that it is better to combine the vocabularies of all the tasks. Furthermore, the **fine-grained learning boosted the results for coarse-grained** while for some datasets, it is better to predict coarse-grained data from fine-grained layer.

4.0 Conclusion and Error Analysis

For lack of computational capacity, every experiment I did was on semcor data only, and the baseline model came out as the overall best. However, some of the medications showed good promise and results could be so different with addition training corpora and possibly different LSTM features.

It is also important to note that the results (accuracy and loss) of the training (both on training data and on the dev set) do not translate to the result of sense predictions (Model B outperforms Models A and C on training metrics while they both produced better sense predictions), it is then safe to conclude that the solution lies in finding a model that learns predictions ONLY for senses in context, while ignoring unannotated words.

The following are some of the reasons as to why it is difficult to get high prediction scores:

1. Some of the sentences in the corpus are too short to learn context
2. The real problem is to learn the right sense among all the possible senses for the word, however, the model consists of learning the right sense among all the available senses in the corpora, and unannotated words too!

5.0 Result Tables and Charts

	Fine-Grained					**Coarse-Grained (lex | dom)**												
	Babelnet ID's						**LexNames**						**Wordnet Domains**					
	Dev	**Test Datasets**					**Dev**	**Test Datasets**					**Dev**	**Test Datasets**				
	SE3	**SE2**	**SE07**	**SE13**	**SE15**	**ALL**	**SE3**	**SE2**	**SE07**	**SE13**	**SE15**	**ALL**	**SE3**	**SE2**	**SE07**	**SE13**	**SE15**	**ALL**
Model A	63.7	61.7	**56.0**	**59.5**	56.7	**60.7**	**78.1**	77.4	70.3	**70.9**	**72.7**	75.0	**87.1**	88.0	86.8	**74.9**	**81.3**	**83.8**
Model B	**64.3**	61.0	53.0	57.5	53.4	59.5	77.9	77.8	70.0	68.9	70.6	74.3	86.6	**88.2**	85.7	74.2	78.8	83.1
Model C	64.0	**62.0**	54.5	58.6	**57.0**	60.6	77.3	78.8	70.8	70.8	72.2	**75.2**	86.7	**88.2**	87.5	74.0	80.3	83.4

Model C + additive att	56.0	54.6	43.3	52.3	49.8	53.0	68.8	72.0	62.2	63.2	63.1	67.3	82.3	85.3	84.6	72.1	76.2	80.2
Model A + mult. att	61.0	59.6	50.1	58.3	52.8	58.0	74.6	76.1	65.7	68.5	69.4	72.4	85.1	87.5	85.5	74.2	79.3	82.6
Multitask (BN + Lex + Dom); different output vocabs	62.6	60.2	53.6	58.6	55.0	59.3	76.7	78.8	66.1	68.9	71.4	74.2	69.5	63.9	86.4	73.2	62.5	68.7
Multitask (BN + Lex + Dom); single output vocab	61.9	61.4	54.7	57.9	54.5	59.3	77.4	**79.2**	67.7	70.3	71.6	74.9	70.5	63.7	**88.8**	74.4	63.7	69.5
	Coarse-grained predictions from fine-grained layer						76.5	78.3	**71.2**	69.7	69.7	74.3	85.4	87.2	85.5	74.0	78.8	82.5

Table 2. Table of results

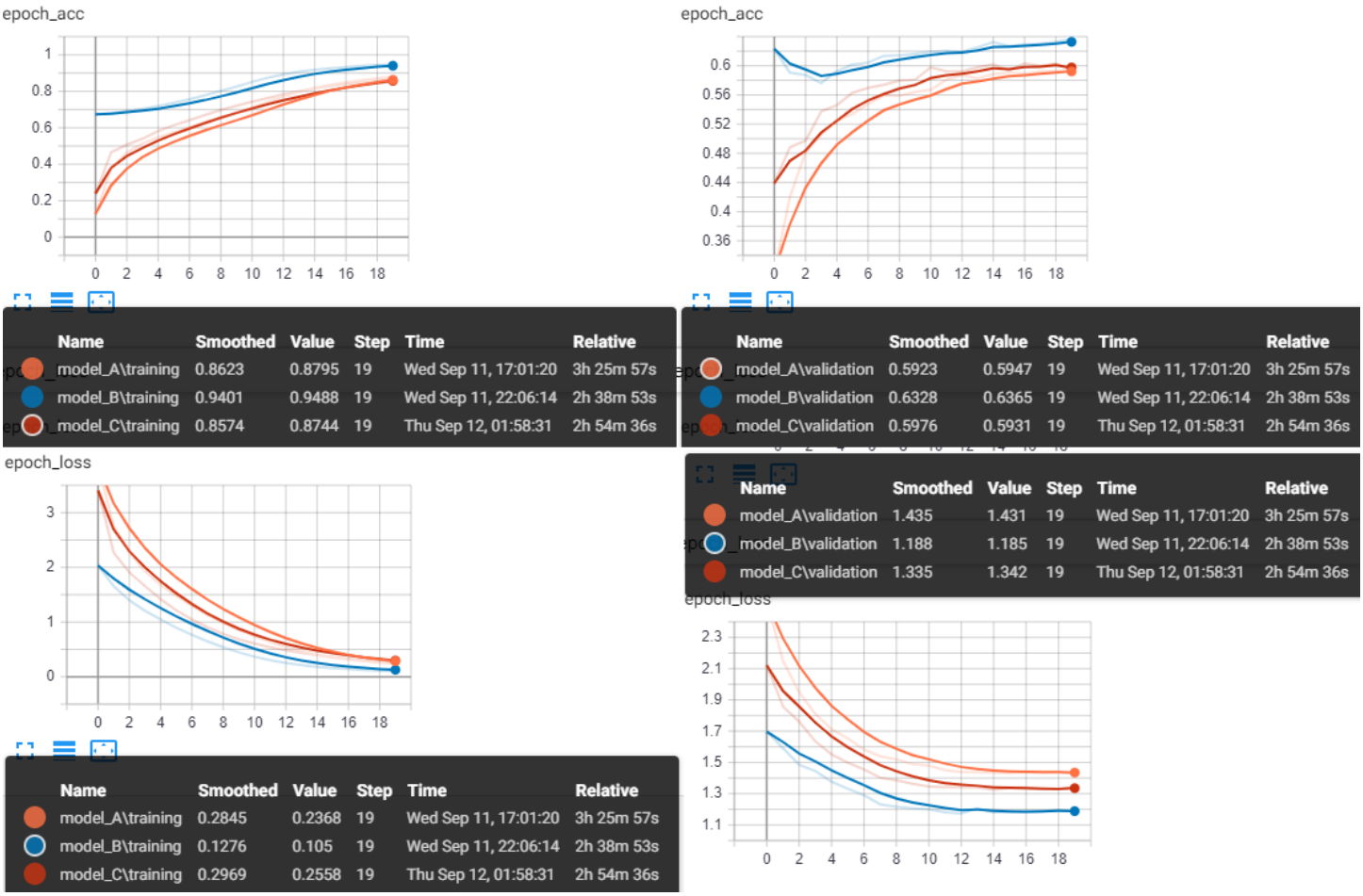

Fig. 4. Model B has the highest accuracy on both training and development data

Fig 5. Multiplicative attention outperforming additive attention in all metrics

6.o References

1. Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli. 2017. Neural Sequence Learning Models for Word Sense Disambiguation
2. Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, Eric Altendorf. 2015. Semi-supervised Word Sense Disambiguation with Neural Models
3. Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang and Zhifang Sui. 2018. Incorporating Glosses into Neural Word Sense Disambiguation
4. Stefano Melacci, Achille Globo, Leonardo Rigutini. 2018. Enhancing Modern Supervised Word Sense Disambiguation Models by Semantic Lexical Resources
5. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. Efficient Estimation of Word
6. https://pypi.org/project/keras-self-attention/